materials and electronics have solved in other contexts. (As one example, a version of Apple Computer's iPod music player uses touch-sensitive buttons instead of electric switches with moving parts—in addition to use a touch-sensitive "dial" control.)

[0214] The device and its software are configured to keep track of the state (e.g. "being touched" or "not being touched") of every one of the sub-sensors on all the device's sensors. (This can be done in a wide variety of ways. For example, the device could poll all the sub-sensors every 1/50th of a second or so to check each sub-sensor's status and update a data array used to track that status. Or the device could be implemented with interrupt mechanisms so that any change in a sub-component results in some software getting executed that updates the memory recording the status of the sub-sensors.)

[0215] Given this configuration, the device gains the ability to determine all of the points along the edge of a given sensor that are being touched (and all the points that are not being touched) at any given moment—to a spatial accuracy and resolution determined by the number and size of the sub-sensor strips and to a time accuracy and resolution determined by the frequency at which the device's electronics and software query or update all of the sub-sensors for their status (which can be quite frequent given the fast processors used in today's mobile electronic devices).

[0216] And given this data of which points (i.e., sub-sensors) along the edge are being touched or not at any given moment, most engineers skilled in the art of device control can come up with algorithms for interpreting the changing status of this data to detect (relatively quick) taps and swipes on some parts of the overall sensor while ignoring other parts of the sensors that appear to be in (relatively long duration) contact with the user's hand (or with other objects). For example, in most contexts, a device designer or sensor engineer could presume that points (i.e., individual sub-sensors) along a touch-sensitive edge that have been in continual contact with something (e.g. the hand of someone holding the device) for relatively long periods of time (e.g. more than 0.5 seconds) can be ignored for the purpose of detecting an upcoming tap or swipe on the device, since taps and swipes involve fairly quick transitions of the status of a sub-sensor from "not being touched" to "being touched" (often changing back to "not being touched" again fairly quickly). If the device temporarily ignores any sub-sensors that have been in a "being touched" state for more than 0.5 seconds (or some appropriate length of time), then the device can watch for tapping by watching for other subs-sensors (perhaps several sub-sensors clustered in a patch about as wide as a finger pad) that suddenly transition from "not being touched" to "being touched" and then transition back to "not being touched" within a short period of time. As described a few pages back, taps, double-taps, triple-taps, and more can be detected using heuristics like this. (Engineers skilled in the art of control input software engineering would typically implement the tap-sensing heuristics to accommodate slight shifts in patches of adjacent sub-sensors that get touched in a multiple tap detection algorithm.)

[0217] It is also fairly straightforward for an engineer to develop software (or firmware) that detects when the user is swiping a finger across this new type of sensor, software that distinguishes the swipe from a tap event, and software that determines the start and end positions of the swipe on the sensor as well as the speed of the swipe. For example, if a set of adjacent sub-sensors on the sensor (making up the patch of sub-sensors that is roughly the width of the pad of a person's finger) suddenly transitions from "not being touched" to "being touched", then a brief time later all of those sub-sensors suddenly transition back to "not being touched" and none of the adjacent sub-sensors change state, then the user probably tapped on that patch of sub-sensor. But if sub-sensors next to the patch suddenly register as "being touched" just when sub-sensors at the other end of the patch of sub-sensors transition back to "not being touched", and if a little later, sub-sensors a next to those newly touched sensors transition to "being touched" just when sub-sensors back a ways transition back to "not being touched", then the user is probably swiping a finger along that sensor. (This is analogous to a piano player dragging a hand up the keyboard of a piano: First one patch set of keys gets depressed, and as the user drags their hand up the piano, the next keys up the piano get depressed as previously depressed keys get released. The reader can look at **FIG. 3**-B and **FIG. 3**-A and picture the sub-sensors as piano keys while picturing the full sensor **301** as a piano. The sub-sensors don't move, but they do sense when they are touched.). The direction of the swipe quickly becomes clear from which sub-sensors are getting "pressed" as neighboring sub-sensors get "released". The distance between any two sub-sensors on a given sensor is fixed—determined by the sensor designer. So the speed of a swipe over a given period can be defined as the distance from the middle of the patch of sub-sensors that transitioned to "being touched" at the beginning of that period (or from one end of that patch if the designers prefers) to the middle of the patch of sub-sensors that transitioned to "being touched" at the end of that period (or to the other end of that second patch if the designer prefers), divided by the time elapsed during that period. (Speed equals distance divided by time.)

[0218] Given that capabilities of the new type of sensor defined here, it is fairly straight forward for engineers skilled in the art of control software and electronics (particularly when that experience is related to touch-sensitive components) to craft algorithms for detecting taps, swipes, and other touch-related interactions with the sensors. And device makers can then use this capability to allow users to control the devices in fun and useful new ways.

[0219] Note that these new types of sensors can also be used to detect when a user taps or touches or swipes two or more fingers at two or more spots on a given sensor at the same time (since the sub-sensors detect contact independently). A device designer could use this feature to allow a user to control the device in interesting new ways. For example, one could picture a device maker letting a user play music on a device by "playing chords" on a sensor on the device—simulating playing chords on a piano. Or one could picture enabling new typing mechanisms in which a user presses multiple spots along the surface or edge of a device simultaneously to type words or letters. Creative device designers can think of many other possible uses for this capability.

[0220] A generally representative list of some of the preferred embodiments of devices that make use of this new class of sensors follows.